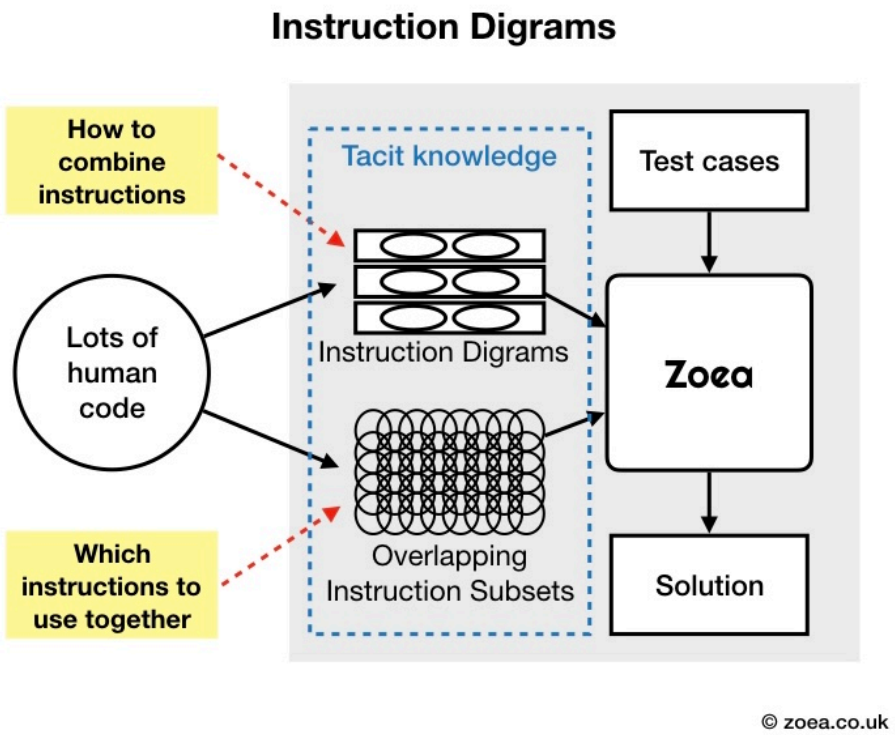
Zoea

Research note 27

Instruction Digrams

Edward McDaid & Sarah McDaid
21 Jun 2023

Zoea has come up with another simple way to make the search space for inductive programming even smaller. This time we use instruction digrams to shrink the problem by another 1000 times.



The ability to code is a remarkable skill. Programming languages typically include hundreds of instructions and these can be combined in astronomical numbers of ways to produce even the simplest programs. Yet if you ask 100 coders to produce a particular program that they've never written before, you don't get 100 completely different solutions. Instead most of them will resemble one of a handful of standard approaches. This is because coders utilise similar programming knowledge.

Programmers accumulate a large quantity of knowledge and experience over the years. Much of this comes from all the code they've written themselves, other peoples code they've been exposed to and all of the bugs they've fixed. They've probably also read books about code, done courses, attended talks, watched videos and so on. Some coders no doubt dream about code.

If you ask a developer to describe their programming knowledge you will get a lot of detail about programming languages - like the keywords and syntax. Beyond this you will get many general rules of thumb and plenty of examples - particularly for different algorithms. However, some of the knowledge that developers use is difficult to articulate and coders probably aren't even aware of it. This is called tacit knowledge.

For one thing, developers don't use all programming language instructions in their programs with equal frequency. A few instructions are really common while most are rarely used. The same thing also goes for pairs of instructions where the output of one instruction is an input to another. We call these pairs instruction digrams.

It turns out that the vast majority of all possible instruction digrams are never used in programs by human developers. We know this because we analysed a huge sample of code and counted all the digrams it contained. Now, most developers probably wouldn't be too surprised to learn this but this fact is really important if you are in the business of generating software automatically.

When you are trying to produce a program from a set of test cases - as in the case of Zoea - there are times when you have to consider all of the possibilities. This can happen when the intent of the specification is not clear. Given that there are hundreds of instructions, blind search of this kind is a huge task. Even a program with only 4 instructions will have more than a billion combinations and this number rises exponentially as program size increases. This is an example of the combinatorial explosion, which has held back AI - and particularly the ability for AI to code - for decades.

Knowing that human developers use relatively few combinations of instructions is really useful. This is like a clue that drastically reduces the number of combinations we have to consider. All we have to do is identify the pairs of instructions that people do use and only

consider those possibilities when we generate code. In fact, this approach typically makes the problem 1000 times smaller - and often more.

Overlapping instruction subsets is a related approach that often reduces the number of combinations by 10 billion times. We can easily combine the subset and digram approaches by only considering digrams where both instructions are present in the same subset. This results is a reduction in the amount of work we need to do - typically by around 13 orders of magnitude. That's the difference between a running time of nearly a million years versus 3 seconds. It also makes it feasible for a very large proportion of the software that people currently write to be generated by Zoea.

Tacit knowledge is shared by many developers but doesn't belong to anyone. Instruction subsets simply capture the common patterns of instruction co-occurrence - that is, which instructions are used together. Instruction digrams go on to describe how these instructions can be assembled. Unlike some other approaches, neither of these misuse anyone's intellectual property.

Learn more at **zoea.co.uk**